Name: Kumar Gaurav

Enrollment NO: 12017002001119

Subject code: CS603

Pg No.: 1

Subject Name: Operating System

Dept/Batch: B.Tech (C.S.E)

Signature: Kumar Gaurav

Answer for CS603

1->2) FIFO: We have to find the first segment large enough to accomodate the incoming segment. If reallocation is not possible & no one segment is large enough select a combination of segments ~~that~~ whose memories are contigious, which are "closest to the first of the last" & which can accomodate the new segment. If ~~relabloca~~ reallocation is possible, rearrange the memory so that the first N segment large enough for the incoming segment are contiguous in memory. Add any leftover space to the free-space list in both cases.

Scanned with CamScanner

Pg no - 2

12017002001119

Kunal Gaurav

LRU : We have to select the segment that has not been used for the longest period of time and that is large enough, adding any leftover space to the free space list. If no one segment is large enough, select a combination of the "oldest" segments that are large enough. If reallocation is available, rearrange the oldest N segments to be contiguous in memory & replace them with the new segment.

Scanned with CamScanner

2→2) Semaphore is simply a variable that is non-negative & shared between threads. A semaphore is signalling mechanism, and a thread that is waiting on a semaphore can be signaled by another thread. It uses two atomic operations, 1) wait & 2) signal for the process synchronization. There are two common kinds of semaphore ie

- Counting semaphore: This type of semaphore uses a count that helps task to be acquired or released numerous times.
- Binary semaphore: The binary semaphore are quite similar to counting semaphore but their value is restricted to 0 & 1. In this type of semaphore the wait operation works only if semaphore = 1, & the signal operation succeeds when semaphore = 0. It is easy to implement than counting semaphore.

Scanned with CamScanner

Structure for operation P() & V() are

```
P (semaphore S){
  while (S==0); /* wait until S=0 */
   S = S-1;
}

V ( semaphore S){
     S = S+1;
  }
```

Implementation of binary semaphore

```
struct semaphore {
  enum value (0,1);
  Queue < process > q;
} P(semaphore S)
{
  if (S.value == 1) {
    S.value = 0;
  }
```

Scanned with CamScanner

1201700200119

Purvi Agarwal

```
    else {
      q.push(P);
      sleep();
    }
  }
v(semaphore s)
{
  if (s.q is empty) {
    s.value = 1;
  }
  else {
  q.pop();
  wakeup();
  }
}
```

Scanned with CamScanner

12017002001119 Kumar Gaurav

3=>1) If we simulate deadlock with a table which is standing on its four legs then we can also simulate four legs with the four conditions which when occurs simultaneously. However it can be prevented by four condition:

- Mutual Exclusion
  - Mutual exclusion from the resource pov is the fact that a resource can never be used by more than one process simultaneously which is fair enough but that is the main reason behind the deadlock.

  However, if we can be able to violate resource behaving in the mutually exclusive manner then the deadlock can be prevented.

  Spooling: By using FCFS, the process doesn't have to wait for the printer & it can continue whatever it was doing.

Scanned with CamScanner

- Hold and Wait
- This condition lies when a process holds a resource & waiting for some other resource to complete its task. Deadlock occurs because there can be more than one process which are holding one resource & waiting for other in the cyclic order.

A process must not wait for any resource once the execution has been started.

!(Hold and wait) = !hold or !wait (negation of hold & wait is, either you dont hold or you dont wait)

- No Preemption
- Deadlock arises due to the fact that a process can't be stopped once it starts. However if we take the resource away from the process which is causing deadlock we can prevent deadlock.

Scanned with CamScanner

- Circular wait
  - To Violate circular wait we can assign a priority no. to each of the resource A process can't request for a lesser priority resource This ensures that not a single process can request a resource which is being utilized by some other process and no cycle is formed.

Condition	Approach	Is practically possible
Mutual Exclusion	Spooling	X
Hold & wait	Request for all the resources	X
NO preemption	Snatch all the resources	X
circular wait	Assign priority to our resources	✓

Scanned with CamScanner

12017CO200119

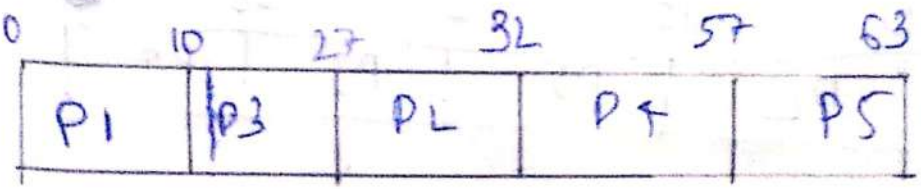
Kunal Gadewar

4⇒ 2)

P_id	A.T	B.T
P1	0	10
P2	1	5
P3	0	17
P4	2	25
P5	7	6

For FCFS

Gantt chart



Scanned with CamScanner

12/7/2021 19

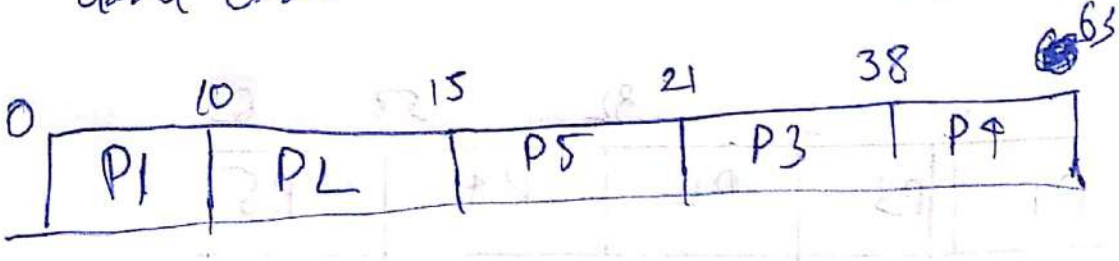
Ram Gaur

	C.T	TAT	WT
P1	10	10-0=10	10-10=0
P2	32	32-1=31	31-5=26
P3	27	27-0=27	27-17=10
P4	57	57-2=55	55-25=30
P5	63	63-7=56	56-6=50

$$AWT = \frac{0+26+10+30+50}{5} = \frac{116}{5} = 23.2$$

For SJF,

Gantt chart



Scanned with CamScanner

pg. no. – 11 | 12017002cs1119 | Kumar Gaurav

	ET	TAT	WT
P_1	10	10 - 0 = 10	10 - 10 = 0
P_2	15	15 - 1 = 14	14 - 5 = 9
P_3	38	38 - 0 = 38	38 - 17 = 21
P_4	63	63 - 2 = 61	61 - 25 = 36
P_5	21	21 - 7 = 14	14 - 6 = 8

$$AWT = \frac{0 + 9 + 21 + 36 + 8}{5} = \underline{14.8}$$

Scanned with CamScanner

1201720020119

Kiran Kumar

5⇒1) File operations: A file is an abstract data type. To defining a file properly, we need to consider the operations that can be performed on files. There are six basic operations with an operating system.

(i) Creating a file: There are two steps necessary for creating a file. First space in the file system must be found for the file. Second an entry for the new file must be made in directory.

ii) Writing a file: To write a file, we make a system call specify about both the name of the file along with the information to be written to the file.

iii) Reading a file: To read a file, we use a system call which specifies the name of the file & where within memory the next block of file should be placed.

Scanned with CamScanner

Repositions inside a file. The directory is then searched for the suitable entry and the 'current file position' pointer is reset to a given value. This file operation is also known as "file seek"

Deleting a file: Deletes the file or directory release all file space so that other files can re-use that space.

Truncating a file: The user may wish for erasing the contents of a file but keep its attributes. Rather than deleting the file and then recreating it, this function allows all attributes to remain unchanged

Scanned with CamScanner

Pg. no - 124 | 120170020011109 | Kiran Ceasar

file type	usual extension	function
executable	exe, com, bin or none	read to run machine language program
Object	obj, o	compiler, machine language, not linked.
batch	bat, sh	Commands to the command interpreter
text	txt, doc	textual data, documents
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

6→1) Race condition: It is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

Example: A simple example is light switch. In some homes there are multiple light switches connected to a common ceiling light. When these types of circuits are used, the switch position become irrelevant. If the light is on, moving either switch from its current position turns the light off. Similarly in the turning off the light. Imagine of what might happen if two people tried to turn on the light using two different switches at exact same time. One might cancel the other or the two actions might trip the circuit breaker.

Scanned with CamScanner

Critical section: It is a segment of code which can be accessed by a signal process at a specific point of time. The section consist of shared data resources that required to be accessed by other processes.

- The entry to the critical section is handled by the wait() & represented as P()
- The exit from a critical section is controlled by signal() function, represented as V()

Example:

```
Process Pi
FLAG[i] = true
while (turn != i) AND
(CS is !free))
{ wait;
}
Critical Section Flag[i] = false
turn = j; // choose another process to go to CS
```

	Flag
P_1	False
P_2	True
P_3	True
P_n	False

Scanned with CamScanner

sleep and wake: If a process is unable to enter a critical section, it is put to sleep/ temporary suspended. Afterwards when the critical section is no longer occupied, the process is woken up.

There is a popular example called producer consumer problem which is most popular in problem simulating sleep & wake mechanism

```
#define N 100
int count = 0
void Producer(void)
    int item;
    while(true)
    produce_item(&item);
    if (count == N) sleep(producer);
    enter_item(item); //puts them in buffer
    count++;
    if (count == 1) wake up(consumer);
```

Scanned with CamScanner

1201700200113

Kiran Gowda

7 => 2) -> i) The content of Matrix need is given below

	A	B	C	D
P0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

ii) The system is in safe state. AND the safe sequence is

P0, P1, P3, P4.

iii) After satisfying P1 request, the system becomes the following state

	ABCD	ABCD	ABCD	ABCD
P0	0012	0012	0000	1100
P1	1420	1750	0330	
P2	1354	2356	1002	
P3	0634 ~~0014~~	0656	~~0020~~ ~~0642~~	
P4	0014	0656	0642	

• The system still in safe state & P0 P2 P1 P3 P4 is safe sequence

Pg No - 19

Date 20/10/2021

Kiran Gowda

8⇒2) Race condition : It is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

Example : A simple example is light switch. In some homes there are multiple light switches connected to a common ceiling light. When these types of circuits are used, the switch position become irrelevant. If the light is on, moving either switch from its current position turns the light off. Similarly in the turning off the light. Imagine if what might happen if two people tried to turn on the light using two different switches at exact same time. One might cancel the other or the two actions might trip the circuit breaker.

Scanned with CamScanner

Critical section: It is a segment of code which can be accessed by a signal process at a specific point of time. The section consist of shared data resources that requires to be accessed by other processes.

- The entry ~~load~~ to the critical section is handled by the wait() & represented as P()
- The exit from a critical section is controlled by signal() function, represented as V()

Example:

```
Process Pi
FLAG[i] = true
while (turn != i) AND
(CS is !free))
{ wait;
}
Critical Section Flag[i] = false
turn = j; // Choose another process to go to CS
```

	Flag
P_1	False
P_2	True
P_3	True
:	
P_n	False

Scanned with CamScanner

Pg No 21 | 120170020011119 | Kumar Ceaun

Sleep and Wake: If a process is unable to enter a critical section, it is put to sleep/ temporary suspended. Afterwards when the critical section is no longer occupied, the process is woken up.

There is a popular example called producer consumer problem which is most popular in problem simulating sleep & wake mechanism

```
#define N 100
int count = 0
void Producer(void)
    int item;
    while(true)
    produce_item(&item);
    if(count == N) sleep(Producer);
    enter_item(item); //puts them in buffer
    count++;
    if(count==1) wake up(consumer);
```

Scanned with CamScanner

9>2) Binding of instruction and data to memory address can be done at

- Compile time: If memory location known as priori, absolute code can be generated must recompile code if starting location changes.
  - MS-DOS.COM format programs are absolute code

- Load time: Must generate relocatable code if memory location is not known at compiling time.

- Execution time: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (eg. base and limit registers

Pg. no. - 23

Base register & limit register are special hardware registers. When a process is given the base register is loaded with the physical location where the process begins in memory. The limit register is loaded with the range of the process. In other words they define the logical address space.

External fragmentation occurs when the memory allocator leaves several contiguous memory blocks b/w portions of allocated memory. For eg: if several memory blocks between portions of allocated ~~Memory~~ in a contiguous fashion but one of the middle block in the line is freed, the free block is fragmented. The block is then available for use by the allocator later if there is need for memory that fits the block but the block is now unable for large memory needs.

Pg. no. - 24 | 12/07/2021 | Kumar Gaurav

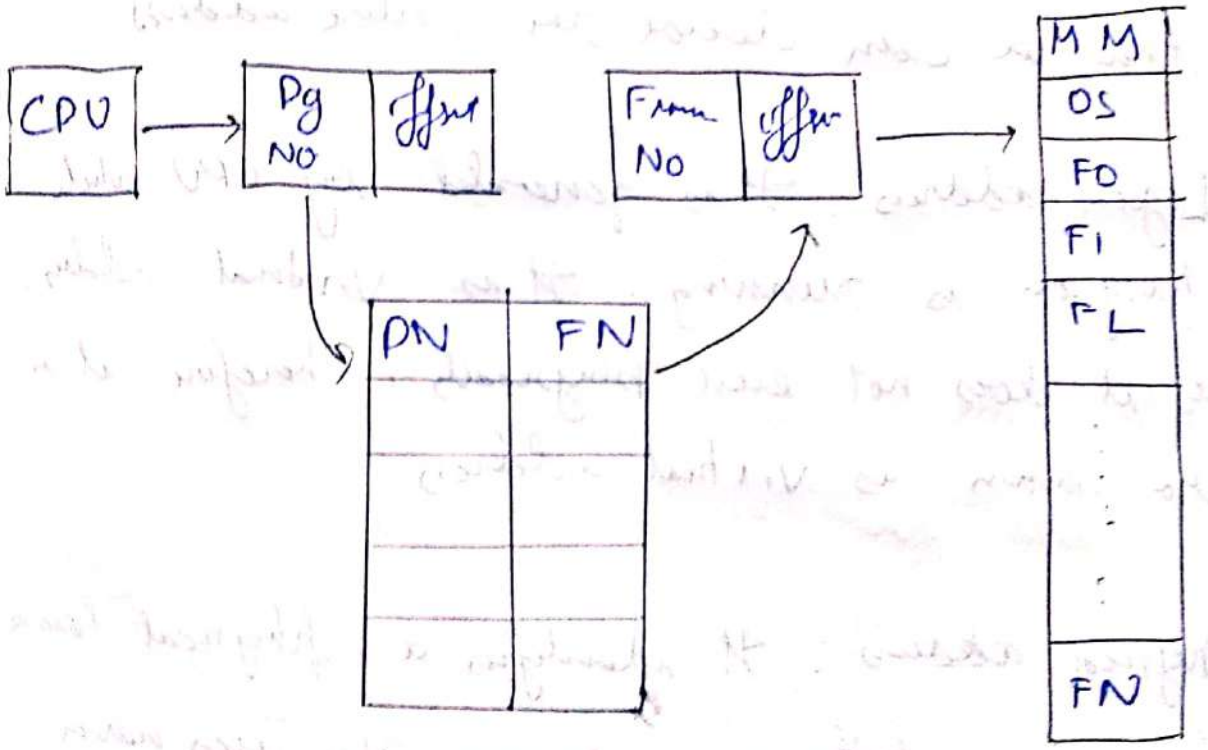
10⇒ 2) Some of the important issues of memory management.

(i) Memory fragmentation (internal & external)

(ii) Memory leak

(iii) Wrong address ~~of~~ assignment.

(iv) Virtual memory management (paging)



When the system allocates a frame to any page, it translates the logical address into a physical address & create entry into the page table to be used throughout the execution of the program.

Scanned with CamScanner

12017002000119

Address translation in contagious memory is an easy task because if we fetch a data from secondary memory & store it into main memory then we fetch the data in a contiguous fashion which means if we know the base address of data then we can decide the entire address.

Logical address: It is generated by CPU when program is running. It is virtual address as it does not exist physically, therefore, it is also known as virtual address.

Physical address: It identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address

Scanned with CamScanner